

# List

List A versatile, feature-rich list component that supports single and multi-selection, integrates with Angular forms, and provides extensive customization options including avatars, icons, and action buttons. Built with accessibility in mind and designed for complex data display scenarios. How to use ■ import { AavaListComponent , AavaListItemsComponent } from "@aava/play-core" ; Note : The List component is standalone and includes all necessary dependencies. The AavaListItemsComponent is used for individual list items with content projection. Basic Usage ■ Simple list implementations with avatars, icons, and basic selection functionality. Multi-Selection ■ Advanced multi-selection capabilities with checkboxes, selection limits, and programmatic control. Multi-Selection Features ■ Checkbox Mode : Visual checkboxes for clear selection indication Selection Limits : Set maximum number of selectable items Programmatic Control : selectAll() , clearSelection() , and selectItems() methods Event Handling : Comprehensive selection change events with detailed information Accessibility ■ WCAG 2.1 AA compliant with comprehensive keyboard navigation and screen reader support. Accessibility Features ■ Keyboard Navigation : Full keyboard support with arrow keys, tab, enter, and escape ARIA Support : Comprehensive ARIA labels, roles, and state announcements Screen Reader : Descriptive labels and status announcements Focus Management : Clear visual focus indicators and logical tab order High Contrast : Enhanced visibility in high contrast modes Testing Checklist : Complete accessibility testing guidelines The List component consists of two main parts: The main list container that handles selection, validation, and form integration. Individual list item wrapper with content projection slots and styling. import { AavaListItemsComponent } from "@aava/play-core" ; @Component ( { selector : "app-list-example" , template : ` <aava-list [items]="items" [multiSelect]="true"> <aava-list-items [selected]="isSelected(item)" [disabled]="item.disabled" [size]="md" (itemClick)="onItemClick(item)" > <!-- Left slot for avatar/icon --> <div left> <aava-avatars [imageUrl]="item.avatar?.imageUrl"></aava-avatars> </div> <!-- Middle slot for content --> <div middle> </div> <!-- Right slot for actions --> <div right> <aava-button [label]="Edit" (userClick)="onEdit(item)" ></aava-button> </div> </aava-list-items> </aava-list> ` , } ) export class ListExampleComponent { // Component implementation } Content Projection Slots ■ [left] : For avatars, icons, or left-aligned content [middle] : For main content like titles and subtitles [right] : For action buttons or right-aligned content Default slot : For any additional content API Reference ■ Inputs ■ Property Type Default Description title string " Title displayed above the list items ListItem[] [] Array of list items to display height string '400px' Height of the list container width string '100%' Width of the list container emptyLabel string 'No items available' Text displayed when list is empty multiSelect boolean false Enable multi-selection mode maxSelections number undefined Maximum number of items that can be selected selectedItemId string | null null Currently selected item ID (single select) selectedItemIds string[] [] Array of selected item IDs (multi-select) showCheckboxes boolean false Show checkboxes for multi-selection selectionMode 'click' | 'checkbox' 'click' Selection interaction mode required boolean false Whether the list

selection is required errorMessage string 'Please select at least one item' Custom error message  
 errorPosition 'top' | 'bottom' 'bottom' Position of error message showErrorImmediately boolean true  
 Show error immediately or wait for touch Property Type Default Description selected boolean false  
 Whether the item is selected disabled boolean false Whether the item is disabled outline boolean  
 false Whether to show outline styling size ListItemSize 'md' Size variant (xs, sm, md, lg) Outputs ■  
 Event Type Description onSelect EventEmitter<ListItem> Emitted when an item is  
 selected onSelectionChanged EventEmitter<ListSelectionEvent> Emitted when selection changes  
 onButtonClick EventEmitter<ListButtonClickEvent> Emitted when an item button is clicked  
 onIconClick EventEmitter<{item: ListItem, event: Event}> Emitted when an item icon is clicked  
 Event Type Description itemClick EventEmitter<void> Emitted when the item is clicked Properties  
 ■ Property Type Description value string | string[] | null Current form value (getter/setter) disabled  
 boolean Whether the component is disabled touched boolean Whether the component has been  
 touched hasError boolean Whether the component has validation errors Methods ■ Method  
 Parameters Return Description selectAll() None void Select all available items (multi-select only)  
 clearSelection() None void Clear all selections selectItems(itemIds: string[]) itemIds: string[] void  
 Select specific items by ID validate() None boolean Manually trigger validation resetValidation()  
 None void Reset validation state hideErrorImmediately() None void Hide error message  
 immediately showErrorImmediatelyMethod() None void Show error message immediately  
 trackByFn(index: number, item: ListItem) index: number, item: ListItem string Track function for  
 efficient rendering trackByButtonFn(index: number, button: ListItemButton) index: number, button:  
 ListItemButton string Track function for button rendering onItemClick(item: ListItem, event?: Event)  
 item: ListItem, event?: Event void Handle item click events onCheckboxChange(item: ListItem,  
 event: Event) item: ListItem, event: Event void Handle checkbox change events  
 onItemButtonClick(item: ListItem, button: ListItemButton, buttonIndex: number, event: Event) item:  
 ListItem, button: ListItemButton, buttonIndex: number, event: Event void Handle button click  
 events onItemIconClick(item: ListItem, event: Event) item: ListItem, event: Event void Handle icon  
 click events hasAvatar(item: ListItem) item: ListItem boolean Check if item has avatar  
 hasIcon(item: ListItem) item: ListItem boolean Check if item has icon hasButtons(item: ListItem)  
 item: ListItem boolean Check if item has buttons isIconClickable(item: ListItem) item: ListItem  
 boolean Check if icon is clickable canSelectMore() None boolean Check if more items can be  
 selected isEmpty() None boolean Check if selection is empty isClickOnActionElement(event:  
 Event) event: Event boolean Check if click is on action element Interfaces ■ ListItem ■ interface  
 ListItem { id : string ; title : string ; subtitle ? : string ; avatar ? : { imageUrl ? : string ; size ? :  
 AvatarSize ; shape ? : AvatarShape ; statusText ? : string ; profileText ? : string ; badgeState ? :  
 BadgeState ; badgeSize ? : BadgeSize ; badgeCount ? : number ; active ? : boolean ;  
 processedanddone ? : boolean ; } ; icon ? : { iconName : string ; color ? : string ; iconColor ? :  
 string ; iconSize ? : number | string ; disabled ? : boolean ; cursor ? : boolean ; } ; buttons ? :  
 ListItemButton [ ] ; disabled ? : boolean ; data ? : any ; } ListItemButton ■ interface ListItemButton  
 { label ? : string ; variant ? : ButtonVariant ; size ? : ButtonSize ; iconName ? : string ; iconColor ? :  
 string ; iconSize ? : number ; iconPosition ? : "left" | "right" | "only" ; disabled ? : boolean ;

processing ? : boolean ; pill ? : boolean ; width ? : string ; height ? : string ; id ? : string ; data ? : any ; } ListSelectionEvent ■ interface ListSelectionEvent { selectedItems : ListItem [ ] ; selectedIds : string [ ] ; lastSelectedItem : ListItem ; } ListButtonClickEvent ■ interface ListButtonClickEvent { item : ListItem ; button : ListItemButton ; buttonIndex : number ; event : Event ; } ListItemSize ■ type ListItemSize = "xs" | "sm" | "md" | "lg" ;

## Design Tokens & Theming

■ AAVA Play List uses semantic design tokens for all surfaces, spacing, radius, and motion. The component exposes scoped override tokens for fine-tuning appearance while maintaining design system consistency.

### Available Design Tokens for List

#### Container Tokens

Token	Purpose	Default Value
--list-container-border-radius	Border radius of list container	Theme-based
--list-container-padding	Padding inside list container	Theme-based
--list-container-gap	Gap between list elements	Theme-based
--list-container-border	Border style for list container	Theme-based

#### Typography Tokens

Token	Purpose	Default Value
--list-background-color	Background color of list	Theme-based
--list-title-color	Color for list title	Theme-based
--list-title-size	Font size for list title	Theme-based
--list-title-weight	Font weight for list title	Theme-based
--list-title-font-family	Font family for list title	Theme-based
--list-item-color	Color for list item text	Theme-based
--list-item-subtitle-color	Color for subtitle text	Theme-based

#### Item Layout Tokens

Token	Purpose	Default Value
--list-items-gap	Gap between list items	Theme-based
--list-item-gap	Gap within list item elements	Theme-based
--list-item-padding	Padding inside list items	Theme-based
--list-item-border-radius	Border radius of list items	Theme-based
--list-item-background	Background color of list items	Theme-based
--list-item-border-color	Border color for list items	Theme-based

#### Selection & State Tokens

Token	Purpose	Default Value
--list-item-active-border	Border style for selected items	Theme-based
--list-active-bg	Background color for selected items	Theme-based
--list-buttons-gap	Gap between action buttons	Theme-based

#### Error & Validation Tokens

Token	Purpose	Default Value
--list-error-text	Color for error messages	Theme-based
--list-error-font-size	Font size for error text	Theme-based
--list-disable-color	Color for disabled elements	Theme-based

```
/* Custom list theming */
.my-compact-list {
  --list-item-padding : 8 px 12 px ;
  --list-items-gap : 4 px ;
  --list-container-border-radius : 4 px ;
}
.my-dense-list {
  --list-title-size : 14 px ;
  --list-item-gap : 8 px ;
  --list-buttons-gap : 4 px ;
}
```

## Best Practices

### Design Guidelines

#### Content Structure

- Use clear, descriptive titles and subtitles for better scanability

#### Avatar Usage

- Provide meaningful avatar content (images or initials) for user identification

#### Icon Integration

- Use appropriate icons that enhance understanding without cluttering

#### Action Buttons

- Limit the number of action buttons per item to maintain clean interface

#### Selection Patterns

- Choose single selection for mutually exclusive choices, multi-selection for independent choices

#### Content Projection

- Use the aava-list-items component for consistent item rendering

#### Slot System

- Utilize left, middle, right, and default slots for flexible content layout

#### Event Handling

- Properly handle click events to avoid conflicts between item selection and button actions

#### Performance

- Use trackBy functions for efficient rendering of large lists

#### State Management

- Leverage the built-in selection state management for consistent behavior

### Accessibility

#### Clear Labeling

- Ensure all interactive elements have descriptive, meaningful labels

#### Keyboard Navigation

- Test complete keyboard navigation flow including arrow keys and activation

#### Screen Reader Support

- Verify proper

announcement of selection changes and item states Color Contrast : Maintain sufficient contrast for all text and interactive elements Focus Management : Provide clear visual focus indicators and logical tab order Performance ■ OnPush Strategy : Component uses OnPush change detection for optimal performance TrackBy Functions : Efficient rendering with custom trackBy functions for large lists Lazy Loading : Consider lazy loading patterns for very large datasets Virtual Scrolling : Implement virtual scrolling for lists with hundreds or thousands of items Event Optimization : Debounce rapid selection changes and optimize event handlers Memory Management : Automatic cleanup of event listeners and references Rendering Optimization : Conditional rendering based on item properties and states Form Integration ■ Validation Strategy : Always validate required selections with clear error messages Form Patterns : Use reactive forms for complex validation scenarios Default Values : Set appropriate default selections for better user experience Reset Behavior : Define clear reset and initial state behavior for forms Cross-Field Validation : Implement proper validation relationships between form fields ControlValueAccessor : Full implementation for seamless form integration Touch Management : Automatic touch state management for validation timing Error Display Control : Programmatic control over error message visibility

```

<aava-list>
  <aava-list-items *ngFor="let profile of userProfiles">
    <div left>
      <aava-avatars
        size="large"
        shape="pill"
        [imageUrl]="sampleImageUrl"
      ></aava-avatars>
    </div>
    <div middle>
      <h4>{{ profile.heading }}</h4>
      <p>{{ profile.description }}</p>
    </div>
    <div right>
      <aava-icon
        [iconName]='arrow-right'
        iconColor="#000000ff"
        iconSize="24"
      ></aava-icon>
    </div>
  </aava-list-items>
</aava-list>

```

---

```
sampleImageUrl = "assets/1.svg";
```

```

userProfiles = [
  {
    id: 1,
    heading: "Heading comes here",
    description: "Description text goes here",
    avatarUrl: "https://randomuser.me/api/portraits/men/1.jpg",
    iconName: "chevron-right",

    button: {
      text: "label",
      variant: "primary" as ButtonVariant,
      color: "#1976d2",
      action: "view_profile",
    },
  },
  {
    id: 2,
    heading: "Heading comes here",
    description: "Description text goes here",
    avatarUrl: "https://randomuser.me/api/portraits/women/2.jpg",
    iconName: "chevron-right",
    button: {
      text: "label",
      variant: "secondary" as ButtonVariant,
      color: "#388e3c",
      action: "contact",
    },
  },
],

```

```
{
  id: 3,
  heading: "Heading comes here",
  description: "Description text goes here",
  avatarUrl: "https://randomuser.me/api/portraits/men/3.jpg",
  iconName: "chevron-right",
  button: {
    text: "label",
    variant: "primary" as ButtonVariant,
    color: "#f57c00",
    action: "view_portfolio",
  },
},
];
```

■ No code found